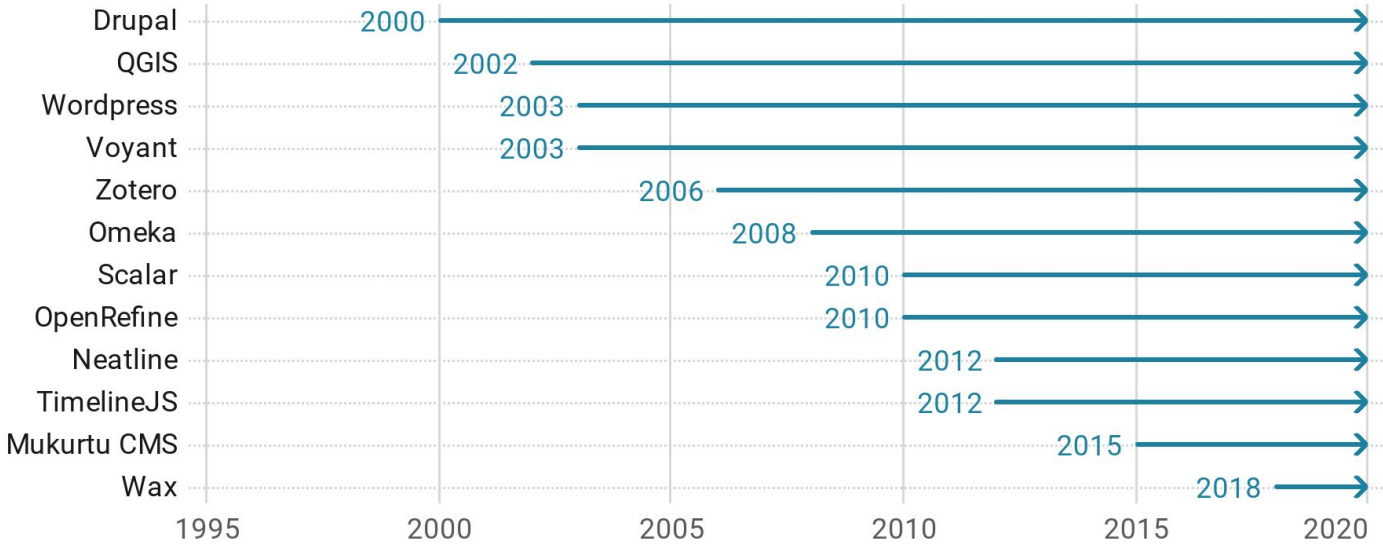
Using Wax and Jekyll to build minimal digital projects

Jo Klein
Cass Wilkinson Saldaña
Angela Zoss (presenting)

DLF 2020 FORUM
25TH ANNIVERSARY

Open source digital scholarship tools over time



Created with Datawrapper

DLF 2020 FORUM
25TH ANNIVERSARY

Why build digital projects minimally?

Rising interest in digital scholarship over recent decades

Challenges:

- Maintenance and ongoing labor
- Learning curve
- Access
- Accessibility
- Autonomy (hosted vs locally managed instances, open source)
- Surveillance
- Justice

DLF 2020 FORUM 25TH ANNIVERSARY

“I prefer to approach minimal computing around the question **“What do we need?”**

...we aim to understand ways of building that could be referred to as **architectures of necessity**…”

“The User, the Learner and the Machines We Make”
by Alex Gil (2015) via Minicomp

DLF 2020 FORUM 25TH ANNIVERSARY

What is Minicomp?

“Minicomp is a collaborative effort in embedding minimal computing principles (e.g. Minimal Maintenance, Minimal Dependencies, and Minimal Connectivity) in digital humanities methods.”

Minicomp Wiki

Minimal computing values in digital projects

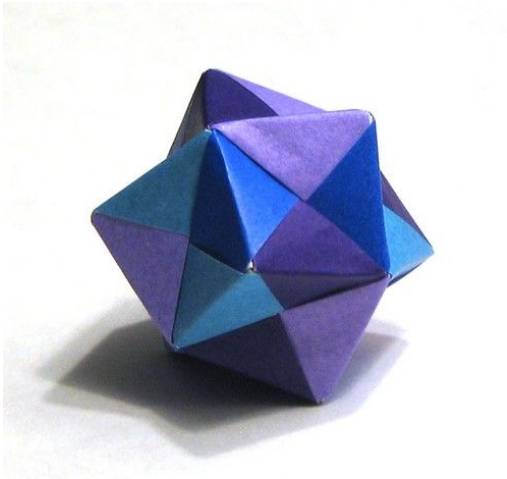
Minimal:

- Maintenance
- Dependencies
- Obsolescence

Maximum:

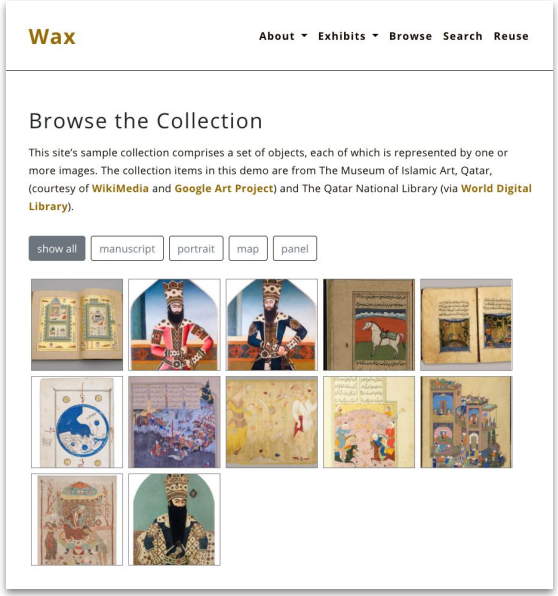
- Access
- Accessibility
- Ephemerality

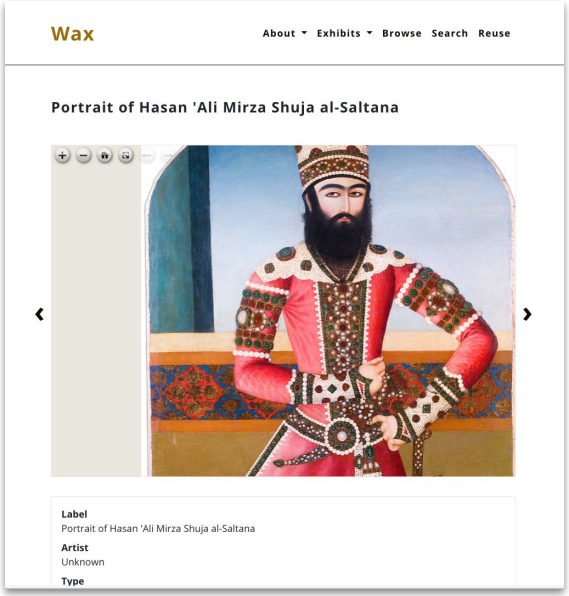
Minimal Definitions (tl;dr version),
Jentery Sayers - 03 Oct 2016



What is Wax?

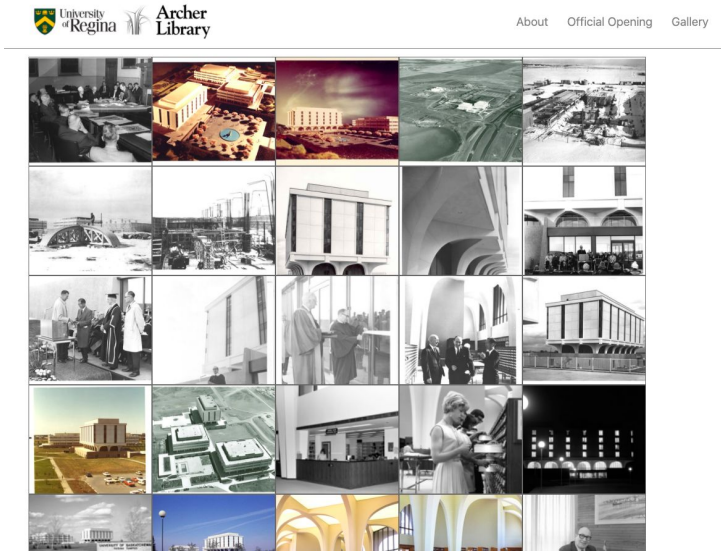
Wax builds on an existing website platform to create minimal digital exhibits.

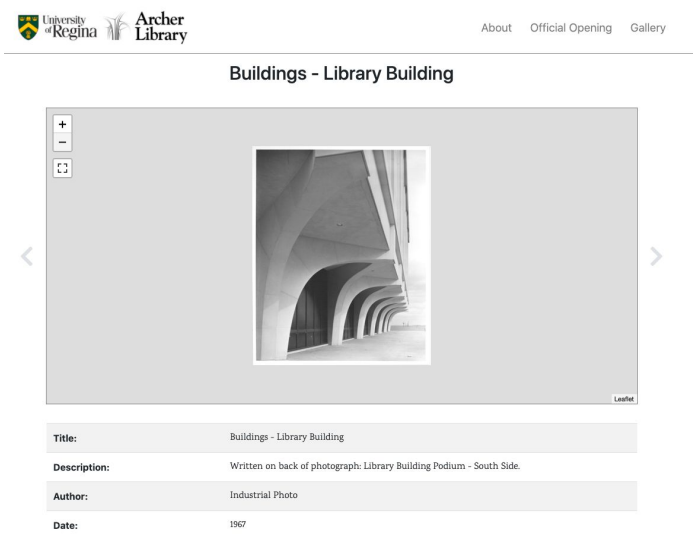




Archer Library - 50th Anniversary Town Hall

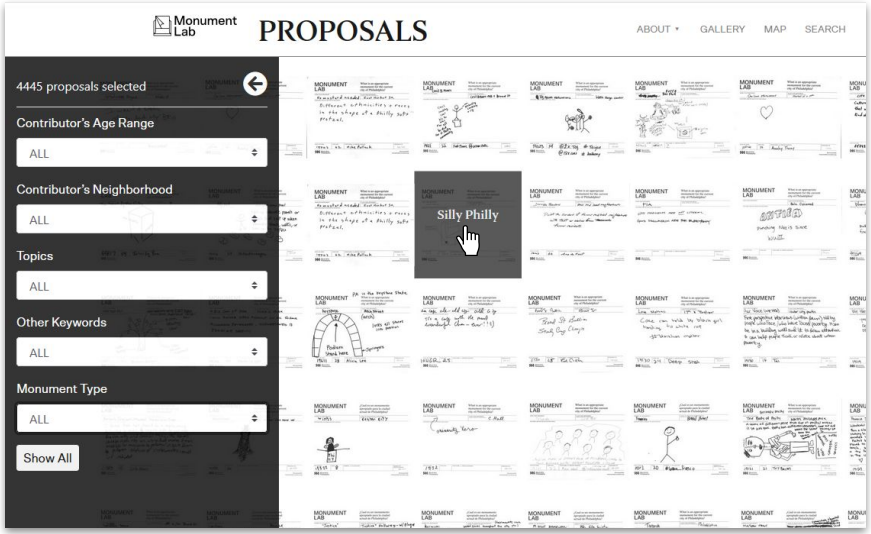
University of Regina - Archer Library





Proposals: Creative Speculations For Philadelphia

Monument Lab & Mural Arts Philadelphia



How Does Wax Work?

First, a note about Jekyll

- Compiles plain text files into a full website
- Built on Ruby
- Has built-in concepts of pages, posts, and (important for Wax) **collections**

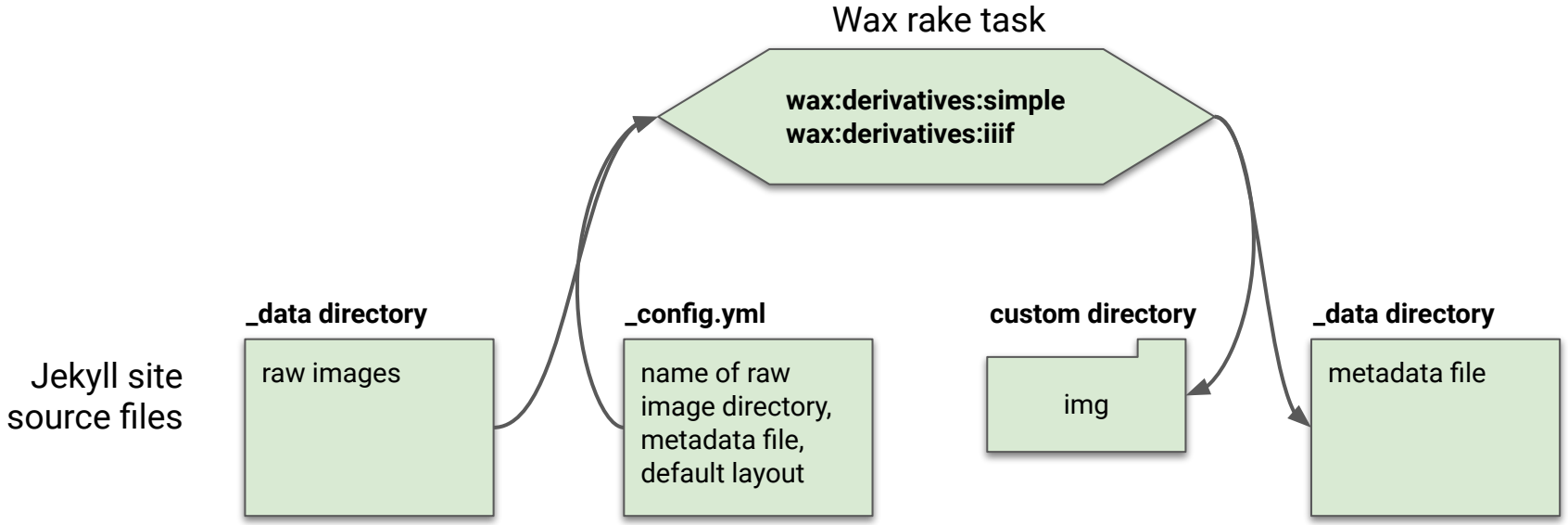
Wax saw Jekyll as a good foundation for digital exhibits.

- Storing objects
- Displaying objects
- Keeping it minimal

Wax helps prepare digital exhibits for Jekyll.

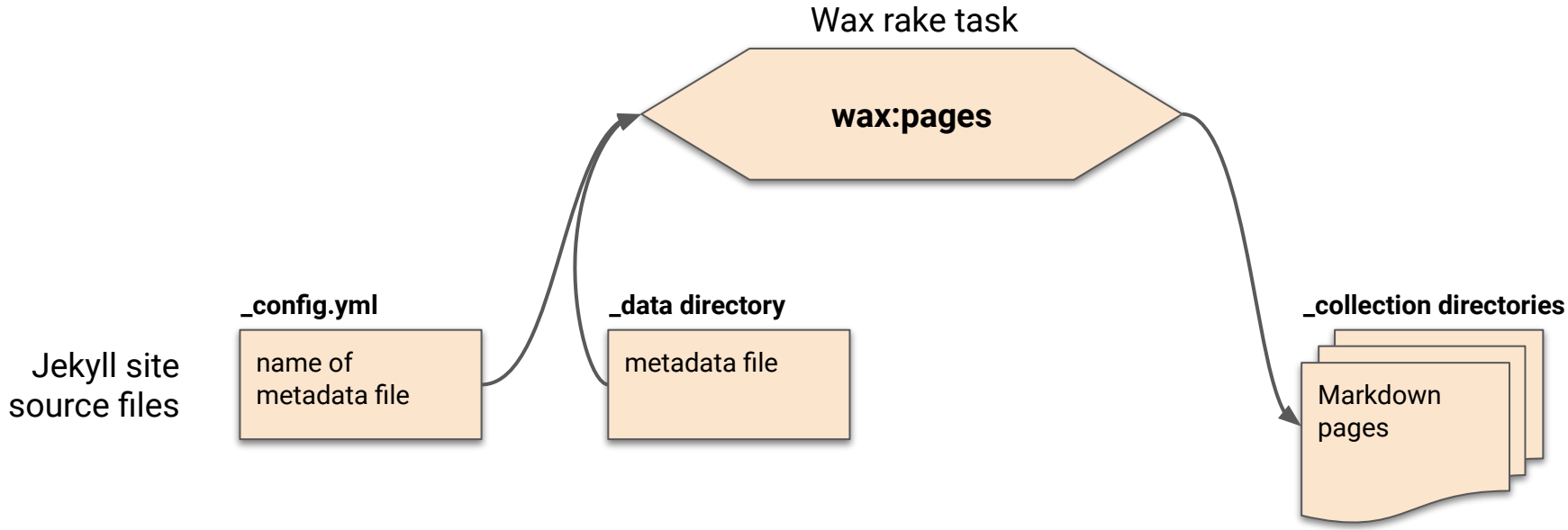
- Processing images
- Generating item pages
- Adding search functionality

Wax image processing

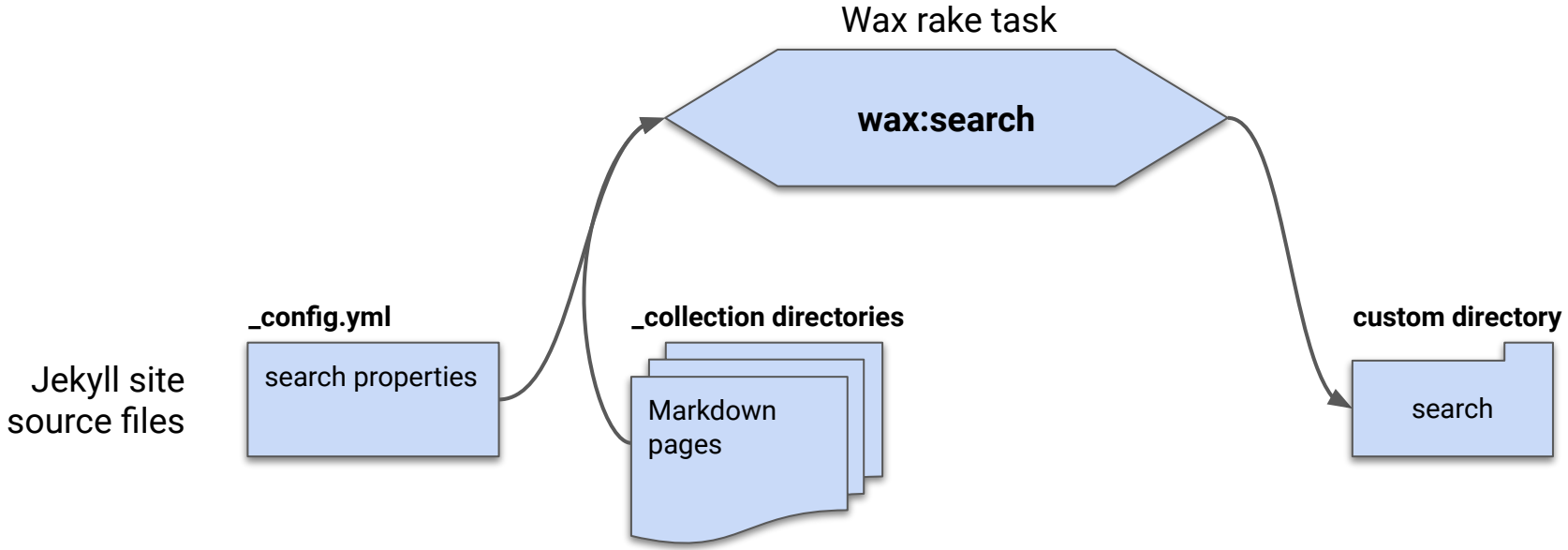


DLF 2020 FORUM 25TH ANNIVERSARY

Wax page generation



Wax search functionality



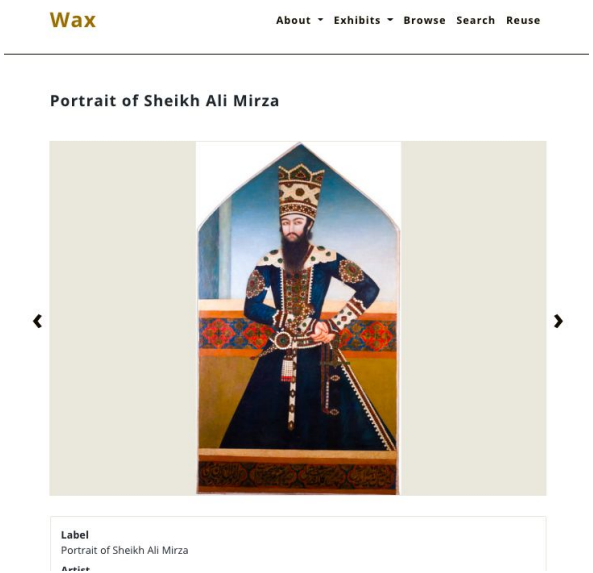
Special pages that come with Wax

“Browse” pages/include

- Repository has a page called “collection”
- Collection page uses an include called “collection_gallery”:
  - pulls in collection metadata
  - builds an image gallery
  - adds filter buttons at the top

“Item” pages/layout

- Wax task creates item pages, including default layout from configuration file
- Repository has layout called “generic_collection_item”
    - image viewer at the top
    - table of metadata at the bottom



Easiest Wax Project

- Download files from the Wax demo repository
- Create a .csv and add to _data folder
- Create raw images and add to _data/collection folder
- Edit configuration file to add collection info, search info, other site information
- Run Wax tasks
- Edit Jekyll pages to update content for current project
- Create a GitHub repository using files
- Set up GitHub Pages for hosting

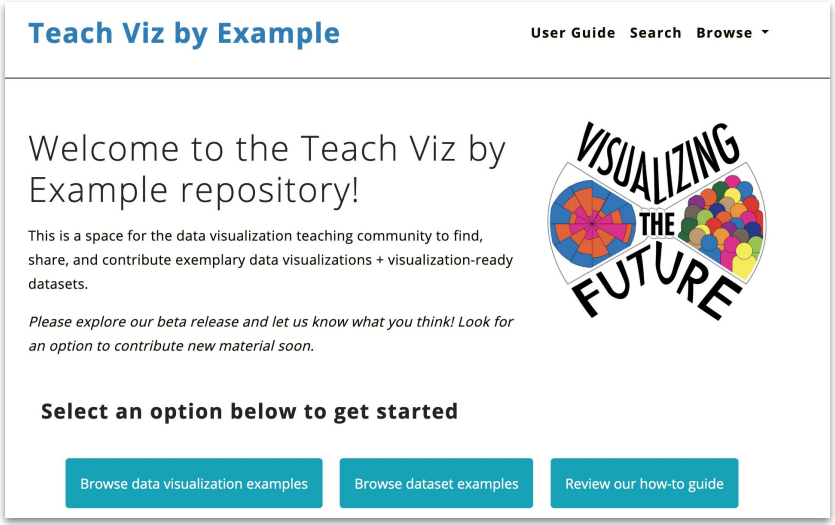
Skills that end up being useful

- Installing software dependencies
- Running and troubleshooting command line tasks
- Creating and maintaining a GitHub repository

DLF 2020 FORUM 25TH ANNIVERSARY

Sample Project: Teaching Viz By Example

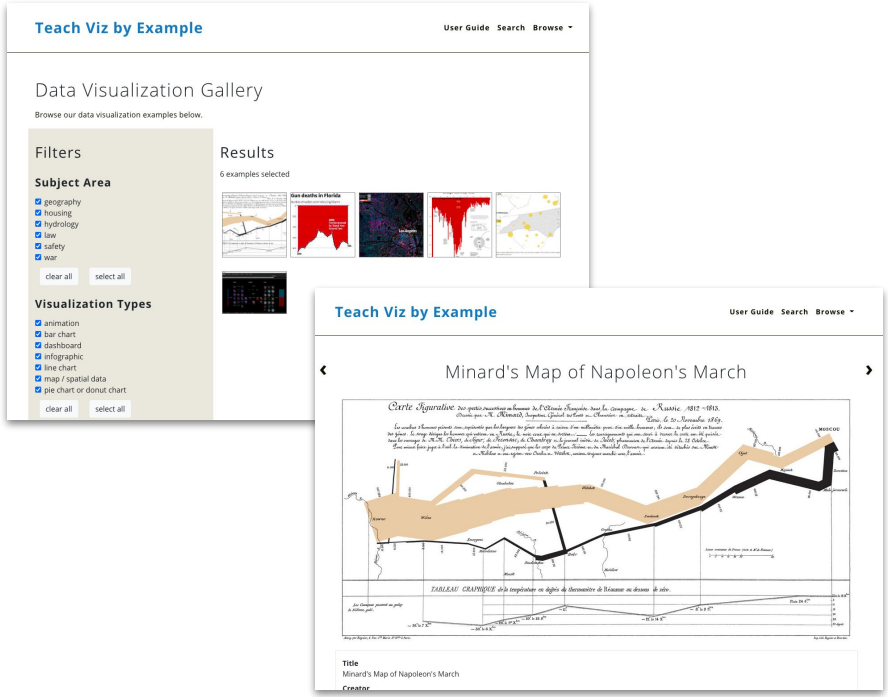
This project was made possible in part by the Institute of Museum and Library Services, RE-73-18-0059-18.



Teach Viz by Example

Why Wax?

- Minimal computing philosophy
- Lightweight, easy to update and maintain
- Host for free on GitHub
- Open to some programming



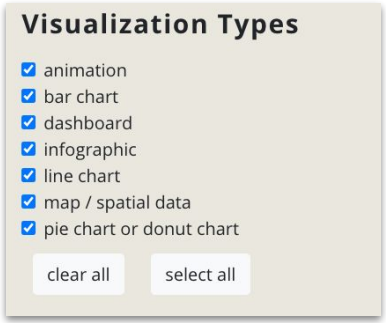
How we are extending Wax

Lists in fields

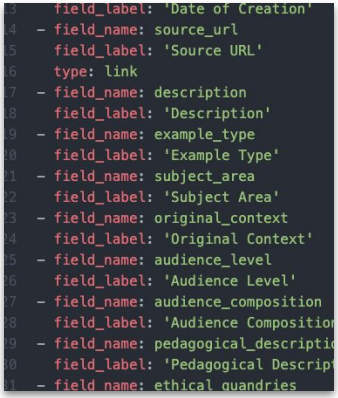
Visualization Types

- line chart
- map / spatial data

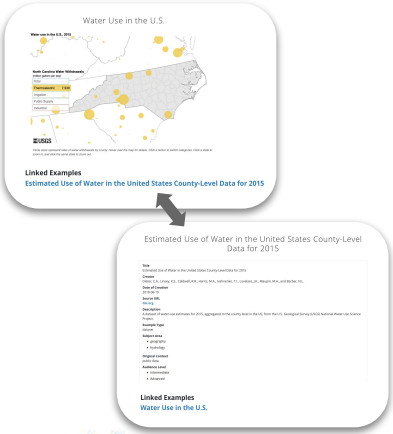
Multi-facet filters



Data dictionary



Linked items



DLF 2020 FORUM
25TH ANNIVERSARY

Pro-Tips

- Cloning is different from downloading! Stick with downloading.
- Wax derivatives tasks update the metadata file, so keep a backup
- Wax doesn’t update metadata if image derivatives already exist, so delete those before rerunning derivatives
- Wax doesn’t regenerate pages, so delete those before rerunning pages task

Resources for learning more

Our grant and project work

- Visualizing the Future
- Examples Repository GitHub
  - Sample Google Apps Script for list processing
  - Python processing script

Writing and Maintaining Code

- Free code editors: Atom, Visual Studio Code
- GitHub Desktop to maintain GitHub repository
- Creating a GitHub Pages site with Jekyll

Advanced but useful

- Bootstrap for website layout
- JQuery for website functionality
- Ruby, if you want to go really deep

More on Jekyll

- Why Use a Static Site Generator?
- Main Jekyll Documentation
- Liquid Template Language

More on Wax

- Wax wiki = ample resources to create your own instance
- Minimal computing for image collections: the case of wax by alex gil, marii nyröp @ DLF2018 Minimal Computing Panel

DLF 2020 FORUM 25TH ANNIVERSARY

Thank you!

Email us at visualizingthefuture@umich.edu

DLF 2020 FORUM 25TH ANNIVERSARY

How we are extending Wax: support for list fields

- Easier to create metadata in spreadsheet, but have some fields that are natural lists
  (e.g., “visualization type”, “audience level”)
- Created python script to take .csv, split lists into arrays, and output JSON
- Have Wax build pages from the JSON instead of the .csv

Title
Minard's Map of Napoleon's March

Creator
Charles Minard

Date of Creation
1869

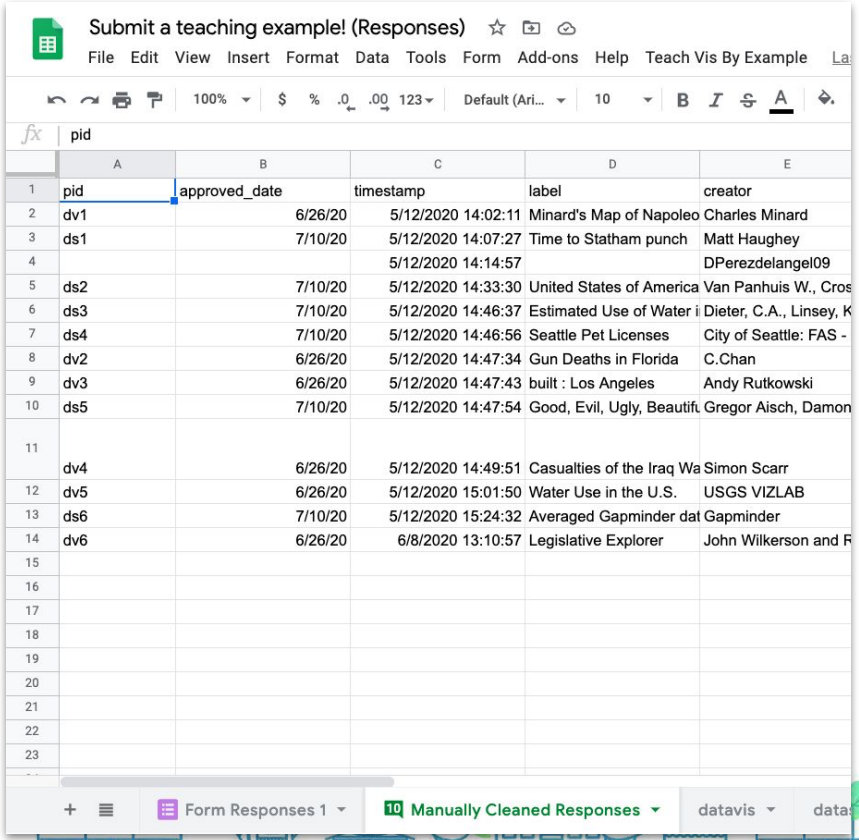
Source URL
upload.wikimedia.org

Description
Charles Minard's visualization depicting Napoleon's march on Moscow

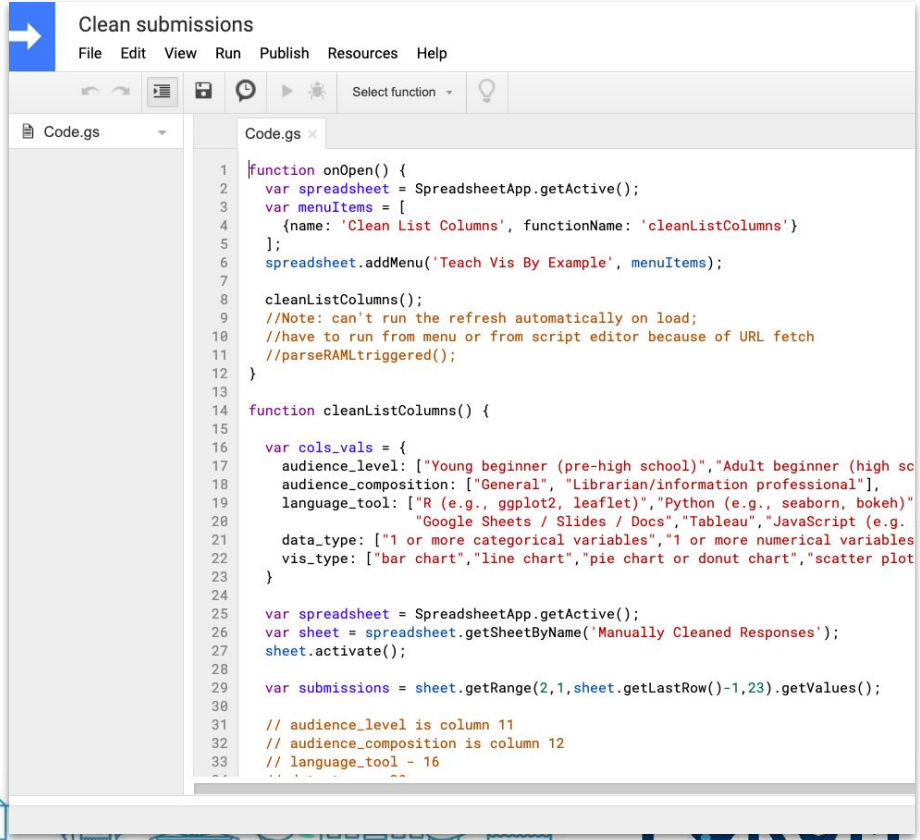
Example Type
data visualization

Visualization Types
- line chart
- map / spatial data

Subject Area
war



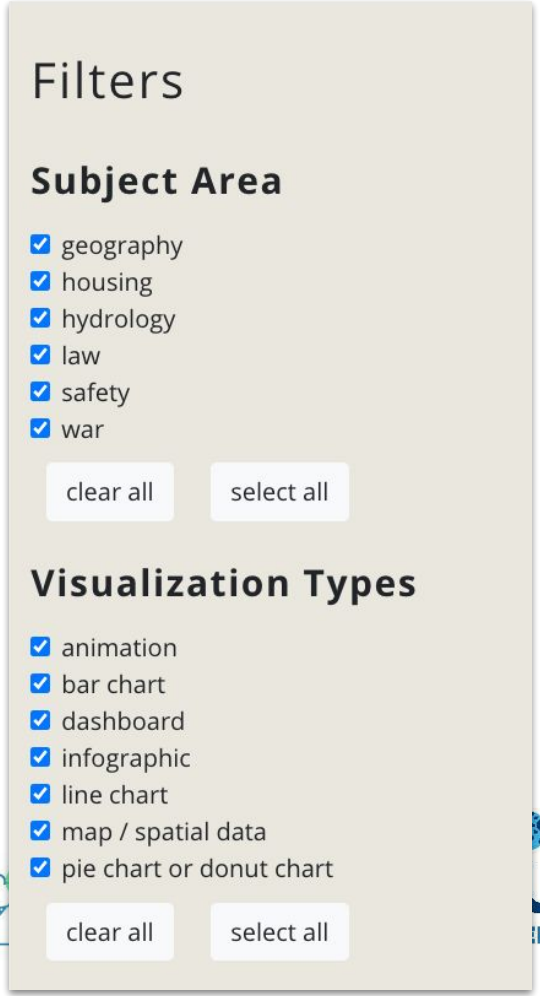
	pid	approved_date	timestamp	label	creator
2	dv1	6/26/20	5/12/2020 14:02:11	Minard's Map of Napoleo	Charles Minard
3	ds1	7/10/20	5/12/2020 14:07:27	Time to Statham punch	Matt Haughey
4			5/12/2020 14:14:57		DPerezdelangel09
5	ds2	7/10/20	5/12/2020 14:33:30	United States of America	Van Panhuis W., Cros
6	ds3	7/10/20	5/12/2020 14:46:37	Estimated Use of Water i	Dieter, C.A., Linsey, K
7	ds4	7/10/20	5/12/2020 14:46:56	Seattle Pet Licenses	City of Seattle: FAS -
8	dv2	6/26/20	5/12/2020 14:47:34	Gun Deaths in Florida	C.Chan
9	dv3	6/26/20	5/12/2020 14:47:43	built : Los Angeles	Andy Rutkowski
10	ds5	7/10/20	5/12/2020 14:47:54	Good, Evil, Ugly, Beautifu	Gregor Aisch, Damon
11	dv4	6/26/20	5/12/2020 14:49:51	Casualties of the Iraq Wa	Simon Scarr
12	dv5	6/26/20	5/12/2020 15:01:50	Water Use in the U.S.	USGS VIZLAB
13	ds6	7/10/20	5/12/2020 15:24:32	Averaged Gapminder dat	Gapminder
14	dv6	6/26/20	6/8/2020 13:10:57	Legislative Explorer	John Wilkerson and R




```
function onOpen() {
  var spreadsheet = SpreadsheetApp.getActive();
  var menuItems = [
    {name: 'Clean List Columns', functionName: 'cleanListColumns'}
  ];
  spreadsheet.addMenu('Teach Vis By Example', menuItems);

  cleanListColumns();
  //Note: can't run the refresh automatically on load;
  //have to run from menu or from script editor because of URL fetch
  //parseRAMLtriggered();
}

function cleanListColumns() {

  var cols_vals = {
    audience_level: ["Young beginner (pre-high school)","Adult beginner (high sc
    audience_composition: ["General", "Librarian/information professional"],
    language_tool: ["R (e.g., ggplot2, leaflet)","Python (e.g., seaborn, bokeh)"
                    "Google Sheets / Slides / Docs","Tableau","JavaScript (e.g.
    data_type: ["1 or more categorical variables","1 or more numerical variables
    vis_type: ["bar chart","line chart","pie chart or donut chart","scatter plot
  }

  var spreadsheet = SpreadsheetApp.getActive();
  var sheet = spreadsheet.getSheetByName('Manually Cleaned Responses');
  sheet.activate();

  var submissions = sheet.getRange(2,1,sheet.getLastRow()-1,23).getValues();

  // audience_level is column 11
  // audience_composition is column 12
  // language_tool - 16
```

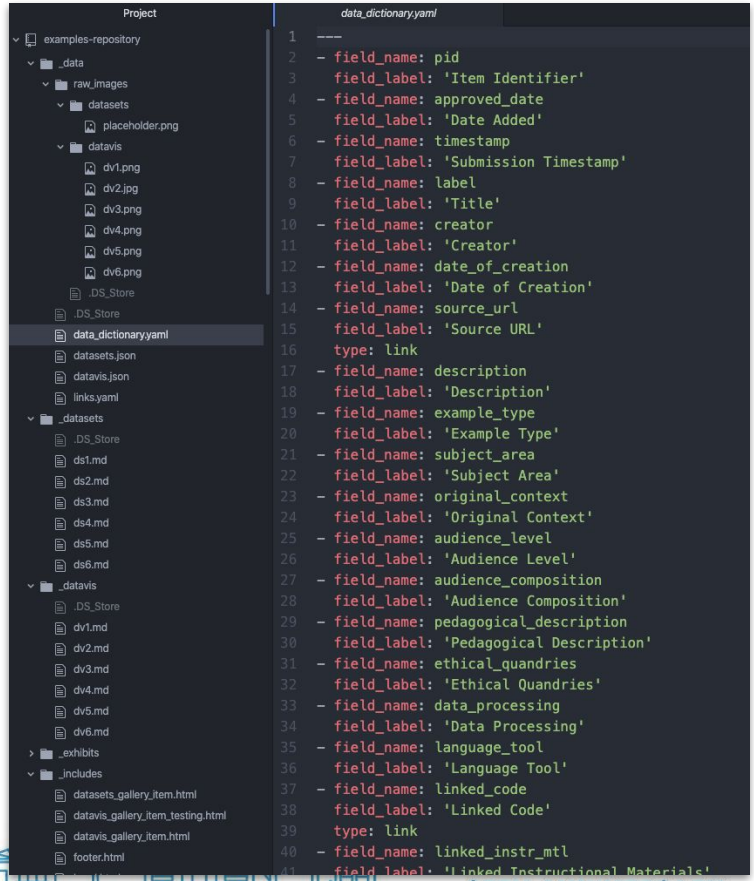
How we are extending Wax: new template for multi-facet filters

- Our use case is really focused on information retrieval, rather than gallery browsing
- Default filter functionality in Wax collection gallery operates for single field, single selection at a time
- Have been working on new checkbox-style filters that allow for selections of multiple values from multiple fields



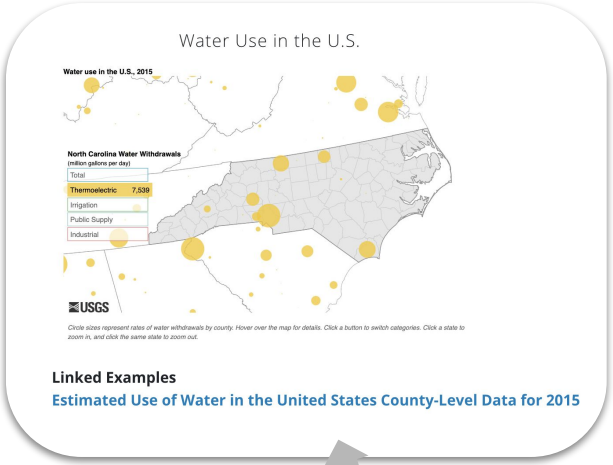
How we are extending Wax: Data dictionary

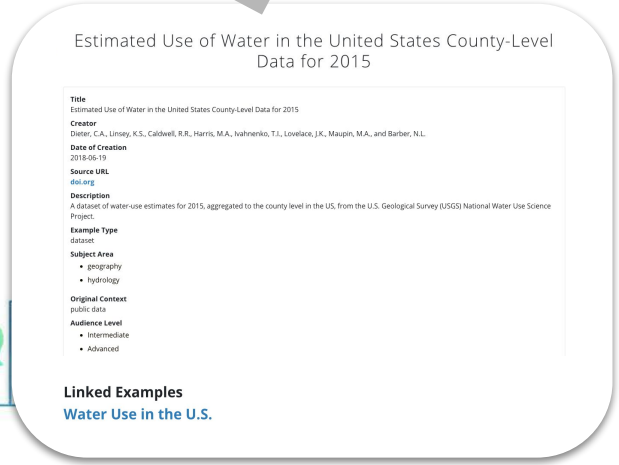
- For some our of website functionality, we needed a way to match up item metadata fields with helpful information about those fields
- Created a data dictionary in YAML format that gets associated with collections and used in different layouts and includes
- Current meta-metadata:
  - field type (list, URL)
  - human-readable name



How we are extending Wax: Connections between examples

- Using the data dictionary, we can specify a metadata field as a special type: “internal”
- Internal fields contain lists of identifiers to other examples in the repository
- On item pages, those internal identifiers get converted to pretty links to the related examples





How we are extending Wax: support for list fields

- Create metadata in spreadsheet (.csv)
- Use python script to split lists into arrays and output JSON
- Have Wax build pages from the JSON instead of the .csv

Title
Minard's Map of Napoleon's March

Creator
Charles Minard

Date of Creation
1869

Source URL
upload.wikimedia.org

Description
Charles Minard's visualization depicting Napoleon's march on Moscow

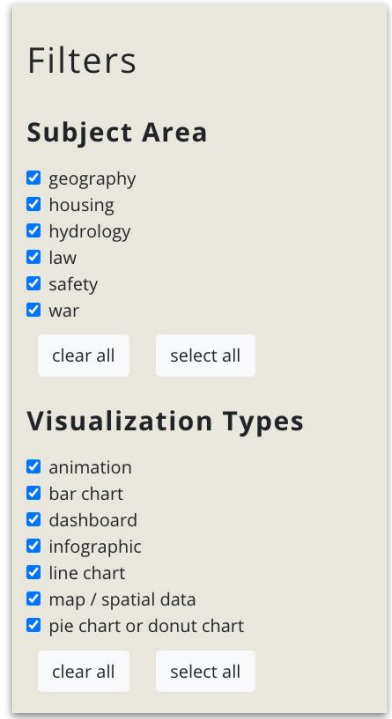
Example Type
data visualization

Visualization Types
- line chart
- map / spatial data

Subject Area
war

How we are extending Wax: new template for multi-facet filters

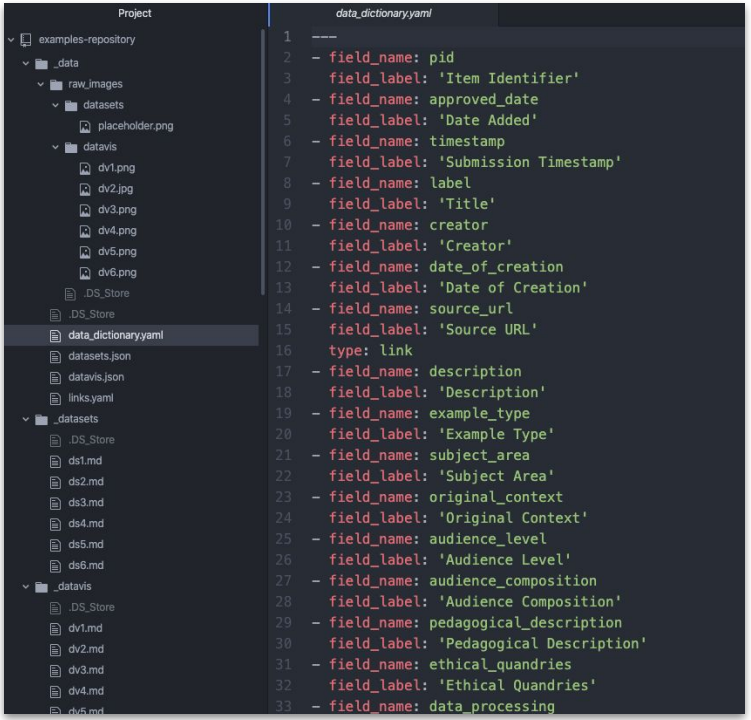
- Default Wax filter: one field, one selection
- Supporting multiple fields, multiple options:
  - support list fields
  - create new page template with new interface elements, filter logic
  - polish user experience (ongoing)



DLF 2020 FORUM 25TH ANNIVERSARY

How we are extending Wax: Data dictionary

- Create data dictionary in YAML format
- Use Jekyll config file to connect collection to data dictionary file
- Current meta-metadata:
  - human-readable name
  - field type (list, URL)



How we are extending Wax: Connections between examples

- Create special field type: “internal”
- Create lists of identifiers of connected examples
- Design item pages to converts IDs to pretty links